5장 우리나라 인구 소멸 위기 지역 분석

1. 목표 명확히 하기
2. 인구 데이터 확보하고 정리하기
3. 인구 소멸 위기 지역 계산하고 데이터 정리하기
4. 대한민국 지도 그리는 방법에 대한 소개
5. 지도 시각화를 위해 지역별 고유 ID 만들기
6. Cartogram으로 우리나라 지도 만들기
7. 인구 현황 및 인구 소멸 지역 확인하기
8. 인구 현황에서 여성 인구 비율 확인하기
9. Folium에서 인구 소멸 위기 지역 표현하기
10. 소감

이름 : 구명회
학번 : 20154215
학과 : 컴퓨터 공학과

1. 목표 명확히 하기

In [1]:
```
# 5-1 목표 명확히 하기
# 인구 소멸 지역의 정의
# 65세 이상 노인 인구와 20~39세 여성 인구를 비교해서
# 젊은 여성 인구가 노인 인구의 절반에 미달할 경우 인구 소멸 지역으로 분류
# 1) 20~30대 여성 인구수를 파악
# 2) 65세 이상 노인 인구수를 파악
# 3) 인구 소멸 위기 지역인지 파악
# 4) Folium 모듈을 이용해서 한국 지도를 그리는 법 학보
```

2. 인구 데이터 확보하고 정리하기

In [2]:
```
# 5-2 인구 데이터 확보하고 정리하기
# 국가 통계 포털에서 해당 데이터를 다운로드
```

In [3]:
```
# 판다스 모듈과 넘피 모듈을 임포트하고
# 한글 폰트를 설정했습니다.
import pandas as pd
import numpy as np

import platform
import matplotlib.pyplot as plt

%matplotlib inline

path = "c:/Windows/Fonts/malgun.ttf"
from matplotlib import font_manager, rc
if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system... sorry~~~~')

plt.rcParams['axes.unicode_minus'] = False
```

In [4]:
```
# 1) 파일의 두 번째 라인부터 읽습니다.
# 2) 빈 셀에 대해 NaN 처리를 하지 않습니다.
# 3) 일부 컬럼의 이름을 수정했습니다.
# 4) '소계'라는 항목을 삭제합니다.
population = pd.read_excel('../data/05. population_raw_data.xlsx', header=1)
population.fillna(method='pad', inplace=True)

population.rename(columns = {'행정구역(동읍면)별(1)':'광역시도',
                             '행정구역(동읍면)별(2)':'시도',
                             '계':'인구수'}, inplace=True)

population = population[(population['시도'] != '소계')]

population.head()
```

Out[4]:

	광역시도	시도	항목	인구수	20 - 24 세	25 - 29 세	30 - 34 세	35 - 39 세	65 - 69 세	70 - 74 세	75 - 79 세	80 - 84 세	85 - 89 세	90 - 94 세	95 - 99 세	100+
6	서울특별시	종로구	총인구수 (명)	152737.0	11379.0	11891.0	10684	10379.0	7411.0	6636.0	5263	3104.0	1480.0	602.0	234	220.0
7	서울특별시	종로구	남자인구수 (명)	75201.0	5620.0	6181.0	5387	5034.0	3411.0	3009.0	2311	1289.0	506.0	207.0	89	73.0
8	서울특별시	종로구	여자인구수 (명)	77536.0	5759.0	5710.0	5297	5345.0	4000.0	3627.0	2952	1815.0	974.0	395.0	145	147.0
9	서울특별시	중구	총인구수 (명)	125249.0	8216.0	9529.0	10332	10107.0	6399.0	5313.0	4127	2502.0	1260.0	469.0	158	160.0
10	서울특별시	중구	남자인구수 (명)	62204.0	4142.0	4792.0	5192	5221.0	3113.0	2405.0	1752	929.0	414.0	132.0	56	51.0

In [5]:
```
# 1) '항목' 컬럼을 '구분'으로 수정합니다.
# 2) '총인구수(명)', '남자인구수(명)', '여자인구수(명)'과 같이 긴 이름을
#     간단히 반복문(for)을 사용해서 '합계', '남자', '여자'로 바꿨습니다.
# 3) 데이터 처리에서 copy 관련 warning을 피하기 위해 .copy()옵션으로 재지정했습니다.
population.is_copy = False

population.rename(columns = {'항목':'구분'}, inplace=True)

population.loc[population['구분'] == '총인구수 (명)', '구분'] = '합계'
population.loc[population['구분'] == '남자인구수 (명)', '구분'] = '남자'
population.loc[population['구분'] == '여자인구수 (명)', '구분'] = '여자'

population.head()
```

```
C:₩Users₩nerin₩Anaconda3₩lib₩site-packages₩pandas₩core₩generic.py:5079: FutureWarning: Attribute 'is_copy' is deprecated and
will be removed in a future version.
  object.__getattribute__(self, name)
C:₩Users₩nerin₩Anaconda3₩lib₩site-packages₩pandas₩core₩generic.py:5080: FutureWarning: Attribute 'is_copy' is deprecated and
will be removed in a future version.
  return object.__setattr__(self, name, value)
```

Out[5]:

	광역시도	시도	구분	인구수	20 - 24세	25 - 29세	30 - 34세	35 - 39세	65 - 69세	70 - 74세	75 - 79세	80 - 84세	85 - 89세	90 - 94세	95 - 99세	100+
6	서울특별시	종로구	합계	152737.0	11379.0	11891.0	10684	10379.0	7411.0	6636.0	5263	3104.0	1480.0	602.0	234	220.0
7	서울특별시	종로구	남자	75201.0	5620.0	6181.0	5387	5034.0	3411.0	3009.0	2311	1289.0	506.0	207.0	89	73.0
8	서울특별시	종로구	여자	77536.0	5759.0	5710.0	5297	5345.0	4000.0	3627.0	2952	1815.0	974.0	395.0	145	147.0
9	서울특별시	중구	합계	125249.0	8216.0	9529.0	10332	10107.0	6399.0	5313.0	4127	2502.0	1260.0	469.0	158	160.0
10	서울특별시	중구	남자	62204.0	4142.0	4792.0	5192	5221.0	3113.0	2405.0	1752	929.0	414.0	132.0	56	51.0

3. 인구 소멸 위기 지역 계산하고 데이터 정리하기

In [6]:
```
# 5-3 인구 소멸 위기 지역 계산하고 데이터 정리하기
```

In [7]:
```
# 1) 20~30대 여성 인구수를 파악합니다
# 2) 65세 이상 노인 인구수를 파악합니다
population['20-39세'] = population['20 - 24세'] + population['25 - 29세'] + ₩
                        population['30 - 34세'] + population['35 - 39세']

population['65세이상'] = population['65 - 69세'] + population['70 - 74세'] + ₩
                        population['75 - 79세'] + population['80 - 84세'] + ₩
                        population['85 - 89세'] + population['90 - 94세'] + ₩
                        population['95 - 99세'] + population['100+']
population.head()
```

Out[7]:

	광역시도	시도	구분	인구수	20 - 24세	25 - 29세	30 - 34세	35 - 39세	65 - 69세	70 - 74세	75 - 79세	80 - 84세	85 - 89세	90 - 94세	95 - 99세	100+	20-39세	65세이상
6	서울특별시	종로구	합계	152737.0	11379.0	11891.0	10684	10379.0	7411.0	6636.0	5263	3104.0	1480.0	602.0	234	220.0	44333.0	24950.0
7	서울특별시	종로구	남자	75201.0	5620.0	6181.0	5387	5034.0	3411.0	3009.0	2311	1289.0	506.0	207.0	89	73.0	22222.0	10895.0
8	서울특별시	종로구	여자	77536.0	5759.0	5710.0	5297	5345.0	4000.0	3627.0	2952	1815.0	974.0	395.0	145	147.0	22111.0	14055.0
9	서울특별시	중구	합계	125249.0	8216.0	9529.0	10332	10107.0	6399.0	5313.0	4127	2502.0	1260.0	469.0	158	160.0	38184.0	20388.0
10	서울특별시	중구	남자	62204.0	4142.0	4792.0	5192	5221.0	3113.0	2405.0	1752	929.0	414.0	132.0	56	51.0	19347.0	8852.0

In [8]:
```
# 1) 얻어온 컬럼들 중에서 일부만 선택합니다.
# 2) 피벗 테이블을 사용해서
# '합계', '남자', '여자'로 되어 있는 구분을 정리합니다.
# 3) 광역시도와 시도를 index로 잡고,
#    인구수와 20-39세, 65세 이상만 데이터를 가져옵니다.
pop = pd.pivot_table(population,
                     index = ['광역시도', '시도'],
                     columns = ['구분'],
                     values = ['인구수', '20-39세', '65세이상'])
pop.head()
```

Out[8]:

		20-39세			65세이상			인구수		
	구분	남자	여자	합계	남자	여자	합계	남자	여자	합계
광역시도	시도									
강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0
	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0
	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0
	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0
	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0

In [9]:
```
# 인구 소멸 비율을 계산합니다.
pop['소멸비율'] = pop['20-39세','여자'] / (pop['65세이상','합계'] / 2)
pop.head()
```

Out[9]:

		20-39세			65세이상			인구수			소멸비율
	구분	남자	여자	합계	남자	여자	합계	남자	여자	합계	
광역시도	시도										
강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0	1.226041
	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0	0.707314
	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0	1.289738
	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0	0.973990
	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0	1.372647

In [10]:
```
# 소멸비율이라는 컬럼에 인구 소멸 위기 지역을 계산하기 위한 식을 적용하고,
# 이 비율이 1보다 작으면 인구 소멸 위기 지역으로 볼 수 있습니다.
# 계산한 소멸비율이 1이하면 소멸 위기 지역이라고 기록합니다
pop['소멸위기지역'] = pop['소멸비율'] < 1.0
pop.head()
```

Out[10]:

		20-39세			65세이상			인구수			소멸비율	소멸위기지역
	구분	남자	여자	합계	남자	여자	합계	남자	여자	합계		
광역시도	시도											
강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0	1.226041	False
	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0	0.707314	True
	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0	1.289738	False
	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0	0.973990	True
	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0	1.372647	False

In [11]:
```
# 소멸위기지역으로 분류된 리스트를 출력합니다.
pop[pop['소멸위기지역']==True].index.get_level_values(1)
```

Out[11]:
```
Index(['고성군', '삼척시', '양양군', '영월군', '정선군', '평창군', '홍천군', '횡성군', '가평군', '양평군',
       '연천군', '거창군', '고성군', '남해군', '밀양시', '산청군', '의령군', '창녕군', '하동군', '함안군',
       '함양군', '합천군', '고령군', '군위군', '문경시', '봉화군', '상주시', '성주군', '영덕군', '영양군',
       '영주시', '영천시', '예천군', '울릉군', '울진군', '의성군', '청도군', '청송군', '동구', '영도구',
       '강화군', '옹진군', '강진군', '고흥군', '곡성군', '구례군', '담양군', '보성군', '신안군', '영광군',
       '영암군', '완도군', '장성군', '장흥군', '진도군', '함평군', '해남군', '화순군', '고창군', '김제시',
       '남원시', '무주군', '부안군', '순창군', '임실군', '장수군', '정읍시', '진안군', '공주시', '금산군',
       '논산시', '보령시', '부여군', '서천군', '예산군', '청양군', '태안군', '홍성군', '괴산군', '단양군',
       '보은군', '영동군', '옥천군'],
      dtype='object', name='시도')
```

In [12]:
```
pop.head()
```

Out[12]:

		20-39세			65세이상			인구수			소멸비율	소멸위기지역
	구분	남자	여자	합계	남자	여자	합계	남자	여자	합계		
광역시도	시도											
강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0	1.226041	False
	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0	0.707314	True
	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0	1.289738	False
	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0	0.973990	True
	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0	1.372647	False

In [13]:
```
# 피벗 테이블로 구성한 인덱스를 초기화했습니다.
pop.reset_index(inplace=True)
pop.head()
```

Out[13]:

구분	광역시도	시도	20-39세			65세이상			인구수			소멸비율	소멸위기지역
			남자	여자	합계	남자	여자	합계	남자	여자	합계		
0	강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0	1.226041	False
1	강원도	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0	0.707314	True
2	강원도	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0	1.289738	False
3	강원도	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0	0.973990	True
4	강원도	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0	1.372647	False

In [14]:
```
# 다단으로 구성된 컬럼을 합쳤습니다.
# ex) 20-39세 + 남자, 20-39세 + 여자, 20-39세 + 합계
tmp_coloumns = [pop.columns.get_level_values(0)[n] + '
                pop.columns.get_level_values(1)[n]
                for n in range(0,len(pop.columns.get_level_values(0)))]

pop.columns = tmp_coloumns

pop.head()
```

Out[14]:

	광역시도	시도	20-39세남자	20-39세여자	20-39세합계	65세이상남자	65세이상여자	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역
0	강원도	강릉시	26286.0	23098.0	49384.0	15767.0	21912.0	37679.0	106231.0	107615.0	213846.0	1.226041	False
1	강원도	고성군	4494.0	2529.0	7023.0	2900.0	4251.0	7151.0	15899.0	14215.0	30114.0	0.707314	True
2	강원도	동해시	11511.0	9753.0	21264.0	6392.0	8732.0	15124.0	47166.0	46131.0	93297.0	1.289738	False
3	강원도	삼척시	8708.0	7115.0	15823.0	5892.0	8718.0	14610.0	35253.0	34346.0	69599.0	0.973990	True
4	강원도	속초시	9956.0	8752.0	18708.0	5139.0	7613.0	12752.0	40288.0	41505.0	81793.0	1.372647	False

In [15]:
```
# 요약정리 정보를 출력합니다.
pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 13 columns):
광역시도          264 non-null object
시도            264 non-null object
20-39세남자      264 non-null float64
20-39세여자      264 non-null float64
20-39세합계      264 non-null float64
65세이상남자       264 non-null float64
65세이상여자       264 non-null float64
65세이상합계       264 non-null float64
인구수남자         264 non-null float64
인구수여자         264 non-null float64
인구수합계         264 non-null float64
소멸비율          264 non-null float64
소멸위기지역        264 non-null bool
dtypes: bool(1), float64(10), object(2)
memory usage: 25.1+ KB
```

4. 대한민국 지도 그리는 방법에 대한 소개

In [16]:
```
# 5-4 대한민국 지도 그리는 방법에 대한 소개
# Folium 모듈을 사용
# 경계선을 그려주는 json파일을 구하는게 어렵다.
# json 파일을 사용하기 위해 고유 ID를 만들어야 합니다.
```

5. 지도 시각화를 위해 지역별 고유 ID 만들기

In [17]:
```
# 5-5 지도 시각화를 위해 고유 ID 만들기
# 편집된 json파일을 읽을 때 매칭시킬 데이터의 ID와 맞아야 합니다.
# 즉, 지역에 따른 고유 ID가 필요합니다.
```

In [18]:
```
# pop['시도']에 대해 unique를 조사합니다.
pop['시도'].unique()
```

Out[18]:
```
array(['강릉시', '고성군', '동해시', '삼척시', '속초시', '양구군', '양양군', '영월군', '원주시',
       '인제군', '정선군', '철원군', '춘천시', '태백시', '평창군', '홍천군', '화천군', '횡성군',
       '가평군', '고양시', '과천시', '광명시', '광주시', '구리시', '군포시', '권선구', '기흥구',
       '김포시', '남양주시', '단원구', '덕양구', '동두천시', '동안구', '만안구', '부천시', '분당구',
       '상록구', '성남시', '소사구', '수원시', '수정구', '수지구', '시흥시', '안산시', '안성시',
       '안양시', '양주시', '양평군', '여주시', '연천군', '영통구', '오산시', '오정구', '용인시',
       '원미구', '의왕시', '의정부시', '이천시', '일산동구', '일산서구', '장안구', '중원구', '처인구',
       '파주시', '팔달구', '평택시', '포천시', '하남시', '화성시', '거제시', '거창군', '김해시',
       '남해군', '마산합포구', '마산회원구', '밀양시', '사천시', '산청군', '성산구', '양산시', '의령군',
       '의창구', '진주시', '진해구', '창녕군', '창원시', '통영시', '하동군', '함안군', '함양군',
       '합천군', '경산시', '경주시', '고령군', '구미시', '군위군', '김천시', '남구', '문경시',
       '봉화군', '북구', '상주시', '성주군', '안동시', '영덕군', '영양군', '영주시', '영천시',
       '예천군', '울릉군', '울진군', '의성군', '청도군', '청송군', '칠곡군', '포항시', '광산구',
       '동구', '서구', '달서구', '달성군', '수성구', '중구', '대덕구', '유성구', '강서구', '금정구',
       '기장군', '동래구', '부산진구', '사상구', '사하구', '수영구', '연제구', '영도구', '해운대구',
       '강남구', '강동구', '강북구', '관악구', '광진구', '구로구', '금천구', '노원구', '도봉구',
       '동대문구', '동작구', '마포구', '서대문구', '서초구', '성동구', '성북구', '송파구', '양천구',
       '영등포구', '용산구', '은평구', '종로구', '중랑구', '세종특별자치시', '울주군', '강화군', '계양구',
       '남동구', '부평구', '연수구', '옹진군', '강진군', '고흥군', '곡성군', '광양시', '구례군',
       '나주시', '담양군', '목포시', '무안군', '보성군', '순천시', '신안군', '여수시', '영광군',
       '영암군', '완도군', '장성군', '장흥군', '진도군', '함평군', '해남군', '화순군', '고창군',
       '군산시', '김제시', '남원시', '덕진구', '무주군', '부안군', '순창군', '완산구', '완주군',
       '익산시', '임실군', '장수군', '전주시', '정읍시', '진안군', '서귀포시', '제주시', '계룡시',
       '공주시', '금산군', '논산시', '당진시', '동남구', '보령시', '부여군', '서북구', '서산시',
       '서천군', '아산시', '예산군', '천안시', '청양군', '태안군', '홍성군', '괴산군', '단양군',
       '보은군', '상당구', '서원구', '영동군', '옥천군', '음성군', '제천시', '증평군', '진천군',
       '청원구', '청주시', '충주시', '흥덕구'], dtype=object)
```

In [19]:
```
# 고유 아이디를 '광역시도'의 값과 '시도'의 값을 합친것으로 합니다.
# 지도 표기 ID로서의 관결함을 유지하기 위해 '서울 강남'과 같은 형태를 갖습니다.
# 1) 광역시가 아니면서 구를 가지고 있는 시와 그 행정구를 dict형으로 선언합니다
# 2) 광역시도에 있는 이름의 끝 세글자가 '광역시', '특별시', '자치시'로 끝나지 않으면
#    일반 시 혹은 군으로 봅니다.
# 3) 강원도와 경상남도에는 동일한 이름을 가진 '고성군'이 있어서 그것을 처리합니다.
#    세종특별자치시를 '세종'으로 처리하며, 그 외에도 역시 광역시도에서 앞 두글자와
#    시도에서 두 글자인 경우 모두, 아니면 앞 두 글자만 선택해서 고유 ID를 만듭니다.
si_name = [None] * len(pop)

tmp_gu_dict = {'수원':['장안구', '권선구', '팔달구', '영통구'],
               '성남':['수정구', '중원구', '분당구'],
               '안양':['만안구', '동안구'],
               '안산':['상록구', '단원구'],
               '고양':['덕양구', '일산동구', '일산서구'],
               '용인':['처인구', '기흥구', '수지구'],
               '청주':['상당구', '서원구', '흥덕구', '청원구'],
               '천안':['동남구', '서북구'],
               '전주':['완산구', '덕진구'],
               '포항':['남구', '북구'],
               '창원':['의창구', '성산구', '진해구', '마산합포구', '마산회원구'],
               '부천':['오정구', '원미구', '소사구']}

for n in pop.index:
    if pop['광역시도'][n][-3:] not in ['광역시', '특별시', '자치시']:
        if pop['시도'][n][:-1]=='고성' and pop['광역시도'][n]=='강원도':
            si_name[n] = '고성(강원)'
        elif pop['시도'][n][:-1]=='고성' and pop['광역시도'][n]=='경상남도':
            si_name[n] = '고성(경남)'
        else:
            si_name[n] = pop['시도'][n][:-1]

        for keys, values in tmp_gu_dict.items():
            if pop['시도'][n] in values:
                if len(pop['시도'][n])==2:
                    si_name[n] = keys + ' ' + pop['시도'][n]
                elif pop['시도'][n] in ['마산합포구','마산회원구']:
                    si_name[n] = keys + ' ' + pop['시도'][n][2:-1]
                else:
                    si_name[n] = keys + ' ' + pop['시도'][n][:-1]

    elif pop['광역시도'][n] == '세종특별자치시':
        si_name[n] = '세종'

    else:
        if len(pop['시도'][n])==2:
            si_name[n] = pop['광역시도'][n][:2] + ' ' + pop['시도'][n]
        else:
            si_name[n] = pop['광역시도'][n][:2] + ' ' + pop['시도'][n][:-1]
```

In [20]:
```
# 지도 시각화에 사용하기 위한
# 고유 ID(행정구역의 고유한 이름)
# 를 출력합니다.
si_name
```

Out[20]:
```
['강릉',
 '고성(강원)',
 '동해',
 '삼척',
 '속초',
 '양구',
 '양양',
 '영월',
 '원주',
 '인제',
 '정선',
 '철원',
 '춘천',
 '태백',
 '평창',
 '홍천',
 '화천',
 '횡성',
 '가평',
 '고양',
 '과천',
 '광명',
 '광주',
 '구리',
 '군포',
 '수원 권선',
 '용인 기흥',
 '김포',
 '남양주',
 '안산 단원',
 '고양 덕양',
 '동두천',
 '안양 동안',
 '안양 만안',
 '부천',
 '성남 분당',
 '안산 상록',
 '성남',
 '부천 소사',
 '수원',
 '성남 수정',
 '용인 수지',
 '시흥',
 '안산',
 '안성',
 '안양',
```

In [21]:
```
# 결과를 pop의 ID 컬럼에
# 포함시켰습니다.
pop['ID'] = si_name
```

In [22]:
```
# 의미가 없는 컬럼들
# 제거했습니다.
del pop['20-39세남자']
del pop['65세이상남자']
del pop['65세이상여자']

pop.head()
```

Out[22]:

	광역시도	시도	20-39세여자	20-39세합계	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역	ID
0	강원도	강릉시	23098.0	49384.0	37679.0	106231.0	107615.0	213846.0	1.226041	False	강릉
1	강원도	고성군	2529.0	7023.0	7151.0	15899.0	14215.0	30114.0	0.707314	True	고성(강원)
2	강원도	동해시	9753.0	21264.0	15124.0	47166.0	46131.0	93297.0	1.289738	False	동해
3	강원도	삼척시	7115.0	15823.0	14610.0	35253.0	34346.0	69599.0	0.973990	True	삼척
4	강원도	속초시	8752.0	18708.0	12752.0	40288.0	41505.0	81793.0	1.372647	False	속초

6. Cartogram으로 우리나라 지도 만들기

In [23]:
```
# 5-6 Cartogram으로 우리나라 지도 만들기
```

In [24]:
```
# 엑셀 파일을 읽어옵니다.
draw_korea_raw = pd.read_excel('../data/05. draw_korea_raw.xlsx',
                               encoding="EUC-KR")
draw_korea_raw
```

Out[24]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	철원	화천	양구	고성(강원)	NaN	NaN	NaN
1	NaN	NaN	NaN	양주	동두천	연천	포천	의정부	인제	춘천	속초	NaN	NaN	NaN
2	NaN	NaN	NaN	고양 덕양	고양 일산동	서울 도봉	서울 노원	남양주	홍천	횡성	양양	NaN	NaN	NaN
3	NaN	NaN	파주	고양 일산서	김포	서울 강북	서울 성북	가평	구리	하남	정선	강릉	NaN	NaN
4	NaN	NaN	부천 소사	안양 만안	광명	서울 서대문	서울 종로	서울 동대문	서울 중랑	양평	태백	동해	NaN	NaN
5	NaN	인천 강화	부천 원미	안양 동안	서울 은평	서울 마포	서울 중구	서울 성동	서울 강동	여주	원주	삼척	NaN	NaN
6	NaN	인천 서구	부천 오정	시흥	서울 강서	서울 동작	서울 용산	서울 광진	서울 송파	이천	평창	울진	NaN	NaN
7	NaN	인천 동구	인천 계양	안산 상록	서울 양천	서울 관악	서울 서초	성남 중원	과천	광주	영월	영덕	NaN	NaN
8	NaN	NaN	인천 부평	안산 단원	서울 영등포	서울 금천	서울 강남	성남 분당	성남 수정	용인 수지	문경	봉화	NaN	울릉
9	NaN	인천 중구	인천 남구	화성	서울 구로	군포	의왕	수원 영통	용인 기흥	용인 처인	안동	영양	NaN	NaN
10	인천 옹진	인천 연수	인천 남동	오산	안성	수원 권선	수원 장안	제천	예천	영주	구미	청송	포항 북구	NaN
11	태안	아산	천안 동남	천안 서북	평택	음성	수원 팔달	단양	상주	김천	군위	의성	포항 남구	NaN
12	NaN	당진	홍성	예산	공주	진천	충주	청주 흥덕	괴산	칠곡	영천	경산	경주	NaN
13	NaN	서산	보령	청양	세종	대전 대덕	증평	청주 청원	보은	고령	청도	성주	울산 북구	NaN
14	NaN	NaN	부여	논산	계룡	대전 동구	청주 상당	청주 서원	대구 북구	대구 중구	대구 수성	울산 울주	울산 동구	NaN
15	NaN	NaN	서천	금산	대전 유성	대전 중구	옥천	영동	대구 서구	대구 남구	대구 동구	울산 중구	울산 남구	NaN
16	NaN	NaN	군산	익산	대전 서구	무주	거창	합천	대구 달서	대구 달성	부산 금정	부산 동래	부산 기장	NaN
17	NaN	NaN	부안	김제	완주	장수	함양	창녕	밀양	부산 북구	부산 부산진	부산 연제	부산 해운대	NaN
18	NaN	고창	정읍	전주 덕진	진안	남원	진주	의령	부산 강서	부산 사상	부산 동구	부산 중구	NaN	NaN
19	NaN	영광	장성	전주 완산	임실	산청	함안	양산	창원 합포	부산 서구	부산 사하	부산 남구	NaN	NaN
20	NaN	함평	담양	순창	구례	하동	창원 의창	창원 성산	창원 진해	김해	부산 영도	부산 수영	NaN	NaN
21	신안	무안	광주 광산	곡성	화순	광양	사천	창원 회원	통영	NaN	NaN	NaN	NaN	NaN
22	목포	나주	광주 서구	광주 북구	순천	고흥	남해	고성(경남)	거제	NaN	NaN	NaN	NaN	NaN
23	해남	영암	광주 남구	광주 동구	여수	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	진도	강진	장흥	보성	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	완도	NaN	NaN	제주	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN	NaN	서귀포	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [25]:
```
# 각 행정 구역의 화면상 x, y좌표를 얻기 위해
# pivot_table의 반대 개념으로 .stack() 명령을 사용한다.
# index를 재설정하고 컬럼 이름을 수정했습니다.
draw_korea_raw_stacked = pd.DataFrame(draw_korea_raw.stack())
draw_korea_raw_stacked.reset_index(inplace=True)
draw_korea_raw_stacked.rename(columns={'level_0':'y', 'level_1':'x', 0:'ID'},
                              inplace=True)

draw_korea_raw_stacked
```

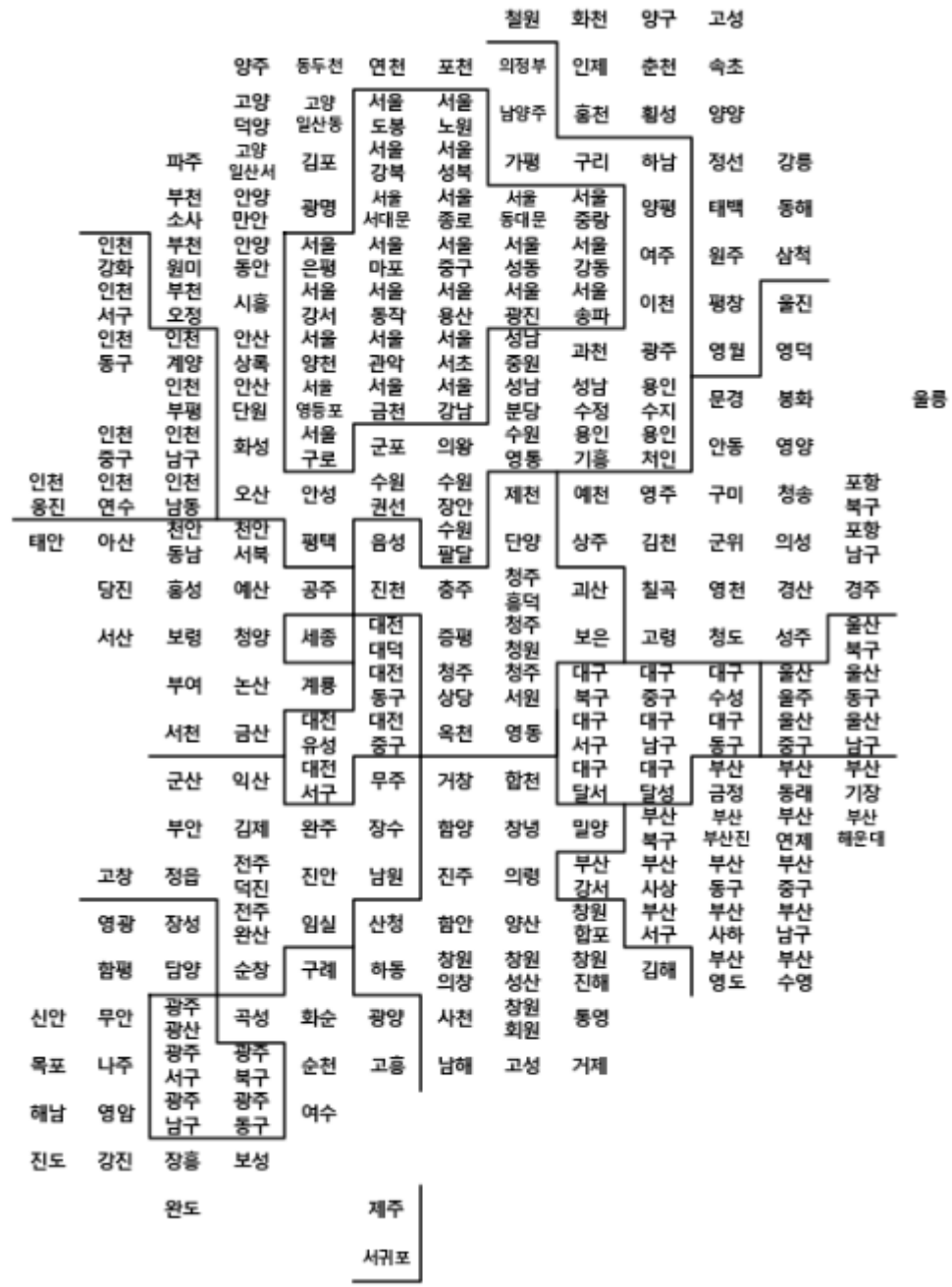
Out[25]:

	y	x	ID
0	0	7	철원
1	0	8	화천
2	0	9	양구
3	0	10	고성(강원)
4	1	3	양주
5	1	4	동두천
6	1	5	연천
7	1	6	포천
8	1	7	의정부
9	1	8	인제
10	1	9	춘천
11	1	10	속초
12	2	3	고양 덕양
13	2	4	고양 일산동

In [26]:
```
# ID 컬럼에서 지도에 표기할때 시 이름 구 이름으로 줄을 나누기 위해 분리했습니다.
# 변수 이름을 변경합니다.
draw_korea = draw_korea_raw_stacked

BORDER_LINES = [
    [(5, 1), (5,2), (7,2), (7,3), (11,3), (11,0)], # 인천
    [(5,4), (5,5), (2,5), (2,7), (4,7), (4,9), (7,9),
     (7,7), (9,7), (9,5), (10,5), (10,4), (5,4)], # 서울
    [(1,7), (1,8), (3,8), (3,10), (10,10), (10,7),
     (12,7), (12,6), (11,6), (11,5), (12, 5), (12,4),
     (11,4), (11,3)], # 경기도
    [(8,10), (8,11), (6,11), (6,12)], # 강원도
    [(12,5), (13,5), (13,4), (14,4), (14,5), (15,5),
     (15,4), (16,4), (16,2)], # 충청북도
    [(16,4), (17,4), (17,5), (16,5), (16,6), (19,6),
     (19,5), (20,5), (20,4), (21,4), (21,3), (19,3), (19,1)], # 전라북도
    [(13,5), (13,6), (16,6)], # 대전시
    [(13,5), (14,5)], #세종시
    [(21,2), (21,3), (22,3), (22,4), (24,4), (24,2), (21,2)], #광주
    [(20,5), (21,5), (21,6), (23,6)], #전라남도
    [(10,8), (12,8), (12,9), (14,9), (14,8), (16,8), (16,6)], #충청북도
    [(14,9), (14,11), (14,12), (13,12), (13,13)], #경상북도
    [(15,8), (17,8), (17,10), (16,10), (16,11), (14,11)], #대구
    [(17,9), (18,9), (18,8), (19,8), (19,9), (20,9), (20,10), (21,10)], #부산
    [(16,11), (16,13)], #울산
#     [(9,14), (9,15)],
    [(27,5), (27,6), (25,6)],
]
```

In [27]:

```
plt.figure(figsize=(8, 11))

# 지역 이름을 표시합니다.
for idx, row in draw_korea.iterrows():

    # 광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시합니다.
    # (중구, 서구)
    if len(row['ID'].split())==2:
        dispname = '{}₩n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
    elif row['ID'][:2]=='고성':
        dispname = '고성'
    else:
        dispname = row['ID']

    # 서대문구, 서귀포시 같이 이름이 3자 이상인 경우에 작은 글자로 표시합니다.
    if len(dispname.splitlines()[-1]) >= 3:
        fontsize, linespacing = 9.5, 1.5
    else:
        fontsize, linespacing = 11, 1.2

    plt.annotate(dispname, (row['x']+0.5, row['y']+0.5), weight='bold',
                 fontsize=fontsize, ha='center', va='center',
                 linespacing=linespacing)

# 시도 경계선을 그렸습니다.
for path in BORDER_LINES:
    ys, xs = zip(*path)
    plt.plot(xs, ys, c='black', lw=1.5)

plt.gca().invert_yaxis()
#plt.gca().set_aspect(1)

plt.axis('off')

plt.tight_layout()
plt.show()
```



In [28]:
```
set(draw_korea['ID'].unique()) - set(pop['ID'].unique())
```

Out[28]:
```
set()
```

In [29]:
```
set(pop['ID'].unique()) - set(draw_korea['ID'].unique())
```

Out[29]:
```
{'고양', '부천', '성남', '수원', '안산', '안양', '용인', '전주', '창원', '천안', '청주', '포항'}
```

In [30]:
```
# pop에 행정구를 가진 시들의 데이터가 더 있는데
# 어차피 지도에서는 표시되지 못하니 삭제합니다.
tmp_list = list(set(pop['ID'].unique()) - set(draw_korea['ID'].unique()))

for tmp in tmp_list:
    pop = pop.drop(pop[pop['ID']==tmp].index)

print(set(pop['ID'].unique()) - set(draw_korea['ID'].unique()))
```

```
set()
```

In [31]:
```
# 다시 pop을 출력합니다
pop.head()
```

Out[31]:

	광역시도	시도	20-39세여자	20-39세합계	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역	ID
0	강원도	강릉시	23098.0	49384.0	37679.0	106231.0	107615.0	213846.0	1.226041	False	강릉
1	강원도	고성군	2529.0	7023.0	7151.0	15899.0	14215.0	30114.0	0.707314	True	고성(강원)
2	강원도	동해시	9753.0	21264.0	15124.0	47166.0	46131.0	93297.0	1.289738	False	동해
3	강원도	삼척시	7115.0	15823.0	14610.0	35253.0	34346.0	69599.0	0.973990	True	삼척
4	강원도	속초시	8752.0	18708.0	12752.0	40288.0	41505.0	81793.0	1.372647	False	속초

In [32]:
```
# pop과 draw_korea의 ID 컬럼이 일치했다고 보고, ID를 key로 merge를 시켰습니다.
pop = pd.merge(pop, draw_korea, how='left', on=['ID'])

pop.head()
```

Out[32]:

	광역시도	시도	20-39세여자	20-39세합계	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역	ID	y	x
0	강원도	강릉시	23098.0	49384.0	37679.0	106231.0	107615.0	213846.0	1.226041	False	강릉	3	11
1	강원도	고성군	2529.0	7023.0	7151.0	15899.0	14215.0	30114.0	0.707314	True	고성(강원)	0	10
2	강원도	동해시	9753.0	21264.0	15124.0	47166.0	46131.0	93297.0	1.289738	False	동해	4	11
3	강원도	삼척시	7115.0	15823.0	14610.0	35253.0	34346.0	69599.0	0.973990	True	삼척	5	11
4	강원도	속초시	8752.0	18708.0	12752.0	40288.0	41505.0	81793.0	1.372647	False	속초	1	10

In [33]:
```
# pop 데이터에서 지도에 표현하고자 하는 데이터가 인구수합계라면
# 이 값들이 아까 만든 각 해당 행정구역에 위치하도록 합니다.
mapdata = pop.pivot_table(index='y', columns='x', values='인구수합계')
masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

mapdata
```

Out[33]:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
y														
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	48013.0	26264.0	24010.0	30114.0	NaN	NaN	NaN
1	NaN	NaN	NaN	205513.0	98277.0	45907.0	154763.0	438457.0	32720.0	280707.0	81793.0	NaN	NaN	NaN
2	NaN	NaN	NaN	446233.0	292612.0	348220.0	567581.0	662154.0	70076.0	45991.0	27218.0	NaN	NaN	NaN
3	NaN	NaN	430781.000000	300839.0	363443.0	327195.0	450355.0	62448.0	193763.0	211101.0	38718.0	213846.0	NaN	NaN
4	NaN	NaN	283793.333333	252353.0	339484.0	314194.0	152737.0	355069.0	411005.0	111367.0	47070.0	93297.0	NaN	NaN
5	NaN	68010.0	283793.333333	345061.0	491476.0	379892.0	125249.0	299259.0	444168.0	111563.0	337979.0	69599.0	NaN	NaN
6	NaN	510733.0	283793.333333	402888.0	595485.0	400997.0	230241.0	357215.0	657831.0	210359.0	43318.0	51738.0	NaN	NaN
7	NaN	71014.0	330284.000000	375857.0	477739.0	506851.0	447192.0	237909.0	63778.0	327723.0	40073.0	39052.0	NaN	NaN
8	NaN	NaN	549716.000000	314002.0	370613.0	235386.0	567115.0	503830.0	232841.0	347833.0	74702.0	33539.0	NaN	10001.0
9	NaN	115249.0	417103.000000	640890.0	417551.0	284890.0	156763.0	340654.0	417163.0	226130.0	168798.0	17713.0	NaN	NaN
10	21351.0	328627.0	530982.000000	208656.0	182896.0	358393.0	296479.0	136517.0	46166.0	109247.0	419891.0	26301.0	272027.0	NaN
11	63900.0	302929.0	258919.000000	359036.0	470832.0	97787.0	198515.0	30503.0	101799.0	142256.0	24171.0	54014.0	244748.0	NaN
12	NaN	166630.0	99971.000000	81339.0	109931.0	69950.0	208350.0	253563.0	38973.0	123199.0	100521.0	258037.0	259452.0	NaN
13	NaN	170788.0	103873.000000	32753.0	243048.0	192688.0	37308.0	190813.0	34221.0	34257.0	43564.0	45205.0	195285.0	NaN
14	NaN	NaN	70187.000000	123213.0	42634.0	234959.0	173701.0	217120.0	440383.0	79712.0	447011.0	219255.0	174514.0	NaN
15	NaN	NaN	56012.000000	54612.0	343222.0	252490.0	52267.0	50552.0	199507.0	156433.0	351352.0	242536.0	340714.0	NaN
16	NaN	NaN	277551.000000	300479.0	491011.0	24949.0	63308.0	48026.0	591891.0	218268.0	244624.0	272745.0	158527.0	NaN
17	NaN	NaN	57005.000000	87782.0	95480.0	23628.0	40241.0	63982.0	108354.0	310202.0	376526.0	207268.0	419853.0	NaN
18	NaN	60597.0	115173.000000	289899.0	26069.0	84188.0	346739.0	28111.0	108909.0	232800.0	89826.0	45208.0	NaN	NaN
19	NaN	55618.0	46104.000000	361845.0	30197.0	36098.0	68937.0	317037.0	181797.0	112973.0	334603.0	278779.0	NaN	NaN
20	NaN	34397.0	47229.000000	29949.0	27412.0	49622.0	252435.0	231571.0	187313.0	529422.0	126362.0	179324.0	NaN	NaN
21	42652.0	82109.0	403049.000000	30400.0	65303.0	155580.0	114912.0	210791.0	138160.0	NaN	NaN	NaN	NaN	NaN
22	237739.0	104376.0	309579.000000	441066.0	278548.0	67656.0	45129.0	54703.0	257183.0	NaN	NaN	NaN	NaN	NaN
23	75121.0	57045.0	219729.000000	95791.0	288988.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	32078.0	37753.0	40669.000000	44469.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	52668.000000	NaN	NaN	470665.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN	NaN	170932.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [34]:
```
# mapdata를 mask형태로 표현합니다
masked_mapdata
```

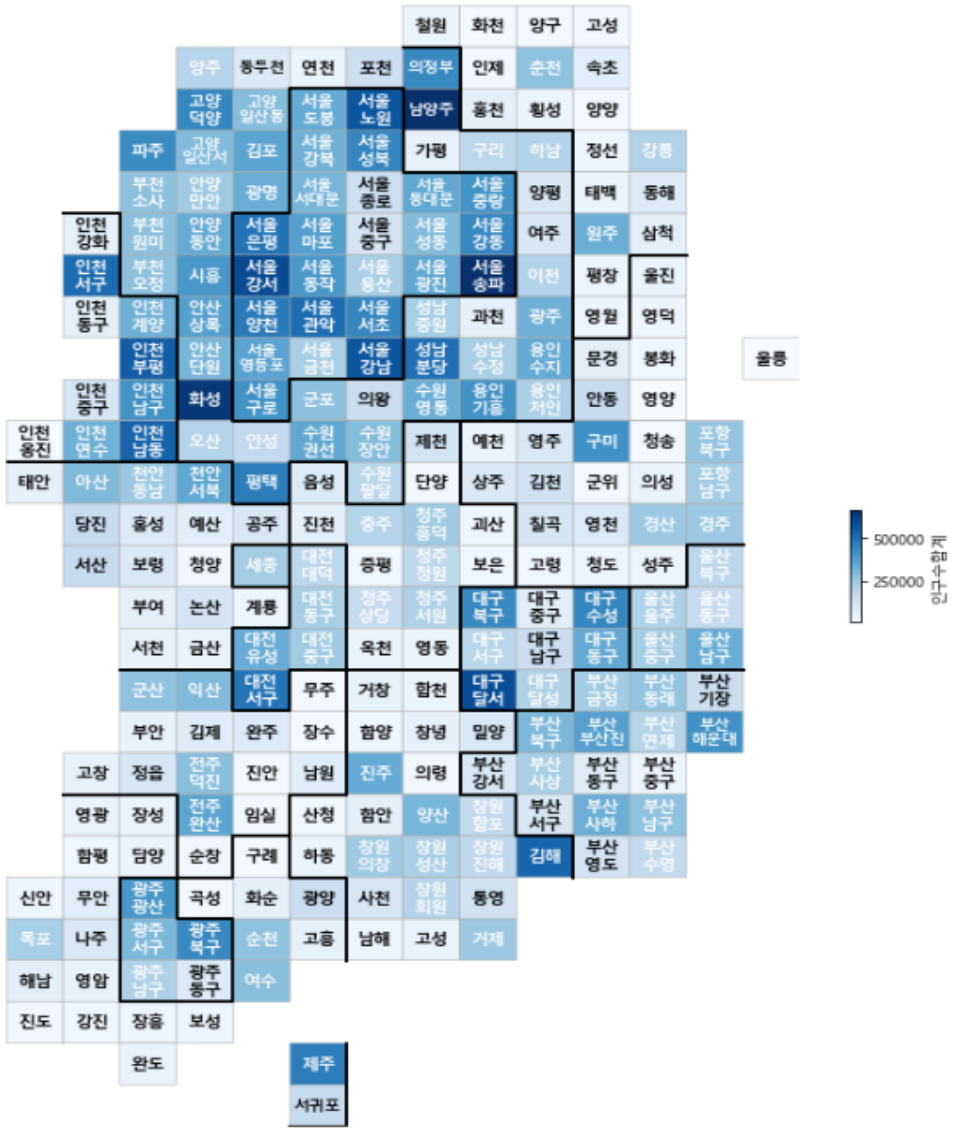
Out[34]:
```
masked_array(
  data=[[--, --, --, --, --, --, --, 48013.0, 26264.0, 24010.0, 30114.0,
         --, --, --],
        [--, --, --, 205513.0, 98277.0, 45907.0, 154763.0, 438457.0,
         32720.0, 280707.0, 81793.0, --, --, --],
        [--, --, --, 446233.0, 292612.0, 348220.0, 567581.0, 662154.0,
         70076.0, 45991.0, 27218.0, --, --, --],
        [--, --, 430781.0, 300839.0, 363443.0, 327195.0, 450355.0,
         62448.0, 193763.0, 211101.0, 38718.0, 213846.0, --, --],
        [--, --, 283793.3333333333, 252353.0, 339484.0, 314194.0,
         152737.0, 355069.0, 411005.0, 111367.0, 47070.0, 93297.0, --,
         --],
        [--, 68010.0, 283793.3333333333, 345061.0, 491476.0, 379892.0,
         125249.0, 299259.0, 444168.0, 111563.0, 337979.0, 69599.0, --,
         --],
        [--, 510733.0, 283793.3333333333, 402888.0, 595485.0, 400997.0,
         230241.0, 357215.0, 657831.0, 210359.0, 43318.0, 51738.0, --,
         --],
        [--, 71014.0, 330284.0, 375857.0, 477739.0, 506851.0, 447192.0,
         237909.0, 63778.0, 327723.0, 40073.0, 39052.0, --, --],
        [--, --, 549716.0, 314002.0, 370613.0, 235386.0, 567115.0,
         503830.0, 232841.0, 347833.0, 74702.0, 33539.0, --, 10001.0],
        [--, 115249.0, 417103.0, 640890.0, 417551.0, 284890.0, 156763.0,
         340654.0, 417163.0, 226130.0, 168798.0, 17713.0, --, --],
        [21351.0, 328627.0, 530982.0, 208656.0, 182896.0, 358393.0,
         296479.0, 136517.0, 46166.0, 109247.0, 419891.0, 26301.0,
         272027.0, --],
        [63900.0, 302929.0, 258919.0, 359036.0, 470832.0, 97787.0,
         198515.0, 30503.0, 101799.0, 142256.0, 24171.0, 54014.0,
         244748.0, --],
        [--, 166630.0, 99971.0, 81339.0, 109931.0, 69950.0, 208350.0,
         253563.0, 38973.0, 123199.0, 100521.0, 258037.0, 259452.0, --],
        [--, 170788.0, 103873.0, 32753.0, 243048.0, 192688.0, 37308.0,
         190813.0, 34221.0, 34257.0, 43564.0, 45205.0, 195285.0, --],
        [--, --, 70187.0, 123213.0, 42634.0, 234959.0, 173701.0,
         217120.0, 440383.0, 79712.0, 447011.0, 219255.0, 174514.0, --],
        [--, --, 56012.0, 54612.0, 343222.0, 252490.0, 52267.0, 50552.0,
         199507.0, 156433.0, 351352.0, 242536.0, 340714.0, --],
        [--, --, 277551.0, 300479.0, 491011.0, 24949.0, 63308.0, 48026.0,
         591891.0, 218268.0, 244624.0, 272745.0, 158527.0, --],
        [--, --, 57005.0, 87782.0, 95480.0, 23628.0, 40241.0, 63982.0,
         108354.0, 310202.0, 376526.0, 207268.0, 419853.0, --],
        [--, 60597.0, 115173.0, 289899.0, 26069.0, 84188.0, 346739.0,
         28111.0, 108909.0, 232800.0, 89826.0, 45208.0, --, --],
```

In [35]:
```
# 헤식님의 메인함수
def drawKorea(targetData, blockedMap, cmapname):
    gamma = 0.75

    whitelabelmin = (max(blockedMap[targetData]) -
                     min(blockedMap[targetData]))*0.25 + #
                     min(blockedMap[targetData])

    datalabel = targetData

    vmin = min(blockedMap[targetData])
    vmax = max(blockedMap[targetData])

    mapdata = blockedMap.pivot_table(index='y', columns='x', values=targetData)
    masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

    plt.figure(figsize=(9, 11))
    plt.pcolor(masked_mapdata, vmin=vmin, vmax=vmax, cmap=cmapname,
               edgecolor='#aaaaaa', linewidth=0.5)

    # 지역 이름을 표시합니다.
    for idx, row in blockedMap.iterrows():
        # 광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시합니다.
        #(중구, 서구)
        if len(row['ID'].split())==2:
            dispname = '{}₩n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
        elif row['ID'][:2]=='고성':
            dispname = '고성'
        else:
            dispname = row['ID']

        # 서대문구, 서귀포시 같이 이름이 3자 이상인 경우에 작은 글자로 표시합니다.
        if len(dispname.splitlines()[-1]) >= 3:
            fontsize, linespacing = 10.0, 1.1
        else:
            fontsize, linespacing = 11, 1.

        annocolor = 'white' if row[targetData] > whitelabelmin else 'black'
        plt.annotate(dispname, (row['x']+0.5, row['y']+0.5), weight='bold',
                     fontsize=fontsize, ha='center', va='center', color=annocolor,
                     linespacing=linespacing)

    # 시도 경계를 그립니다.
    for path in BORDER_LINES:
        ys, xs = zip(*path)
        plt.plot(xs, ys, c='black', lw=2)

    plt.gca().invert_yaxis()
    plt.axis('off')

    cb = plt.colorbar(shrink=.1, aspect=10)
    cb.set_label(datalabel)

    plt.tight_layout()
    plt.show()
```

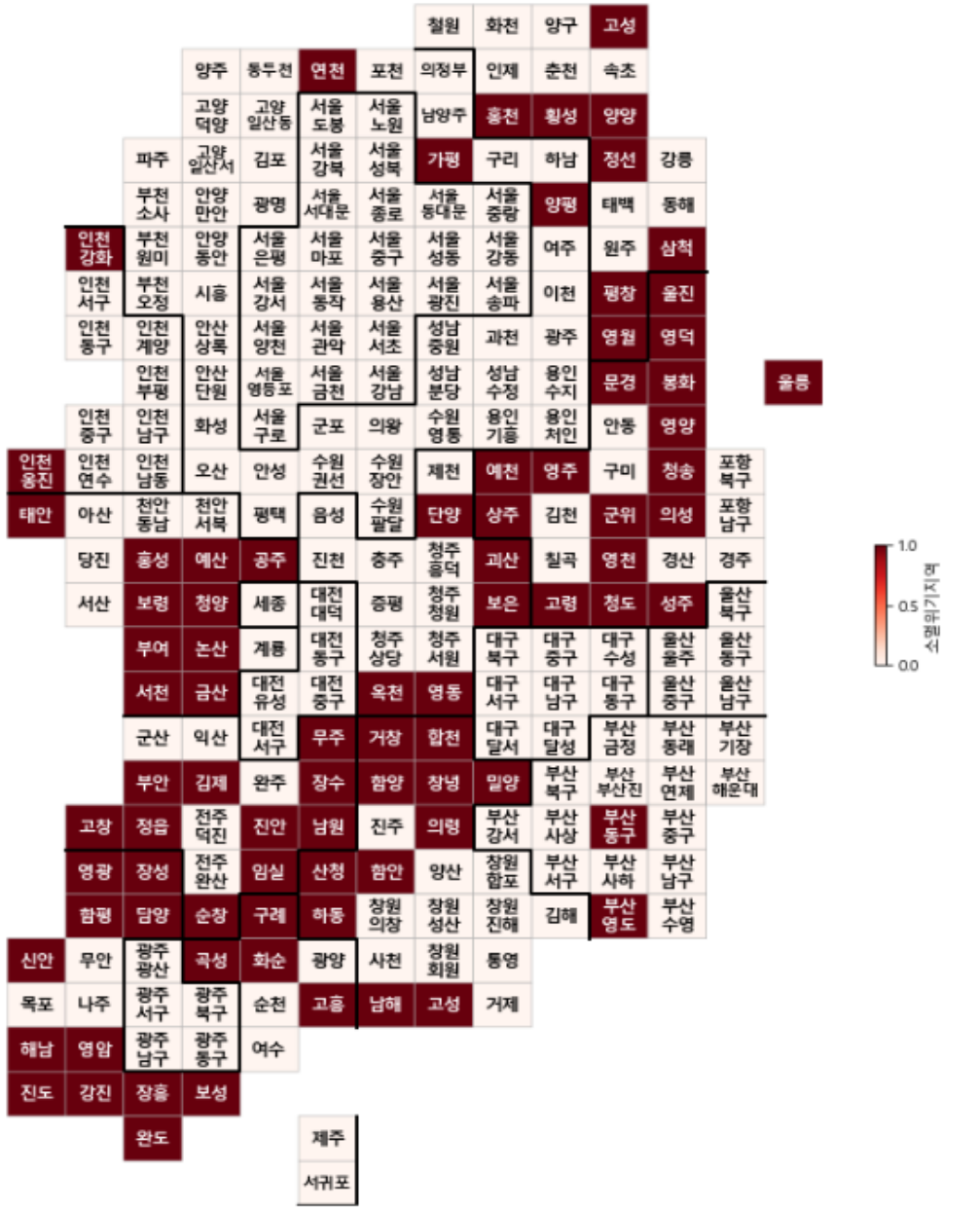
7. 인구 현황 및 인구 소멸 지역 확인하기

In [36]:
```
# 5-7 인구 현황 및 인구 소멸 지역 확인하기
```

In [37]:
```
# 인구수 합계를 그립니다
drawKorea('인구수합계', pop, 'Blues')
```



In [38]:
```
# 인구소멸위기지역을 확인하기 위해서
# bool형으로 설정해둔 소멸위기지역의 값을 0으로 바꿨습니다.
pop['소멸위기지역'] = [1 if con else 0 for con in pop['소멸위기지역']]
drawKorea('소멸위기지역', pop, 'Reds')
```



8. 인구 현황에서 여성 인구 비율 확인하기

In [39]:
```
# 5-8 인구 현황에서 여성 인구 비율 확인하기
```

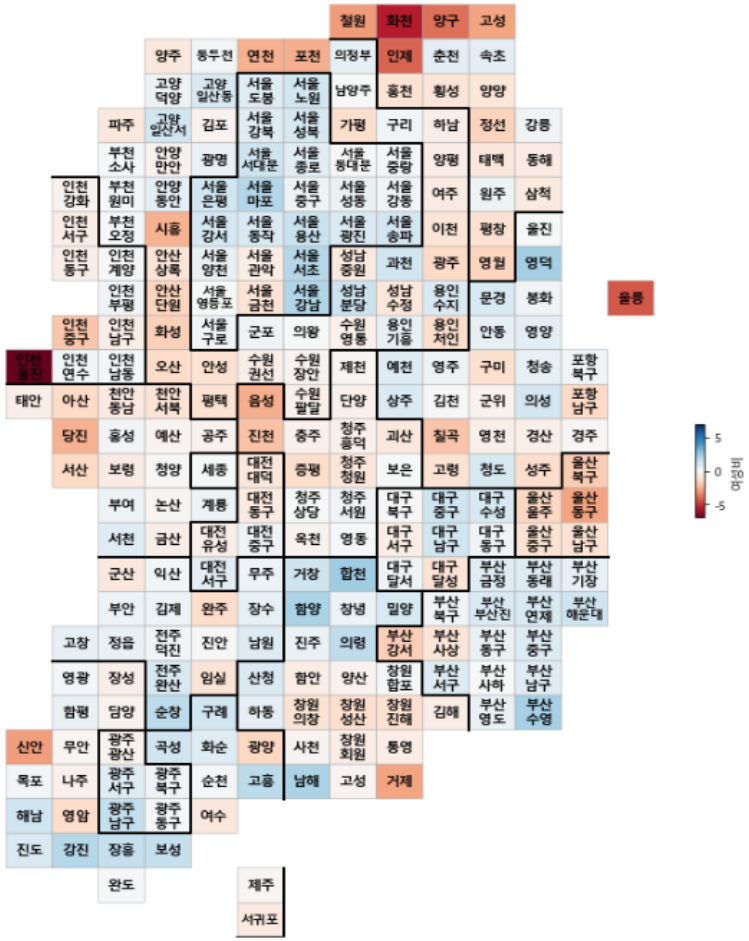
In [40]:
```
# 표현하고자 하는 데이터에 음(-)의 값이 있는지 여부에 따라서
# 일부 설정을 바꿔야 하기 때문에 drawKorea 함수의 내용을 일부 수정합니다.
def drawKorea(targetData, blockedMap, cmapname):
    gamma = 0.75
    whitelabelmin = 20.
    datalabel = targetData

    tmp_max = max([ np.abs(min(blockedMap[targetData])), np.abs(max(blockedMap[targetData]))])
    vmin, vmax = -tmp_max, tmp_max

    mapdata = blockedMap.pivot_table(index='y', columns='x', values=targetData)
    masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

    plt.figure(figsize=(9, 11))
    plt.pcolor(masked_mapdata, vmin=vmin, vmax=vmax, cmap=cmapname, edgecolor='#aaaaaa', linewidth=0.5)

    # 지역 이름을 표시합니다.
    for idx, row in blockedMap.iterrows():
        # 광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시합니다.
        #(중구, 서구)
        if len(row['ID'].split())==2:
            dispname = '{}₩n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
        elif row['ID'][:2]=='고성':
            dispname = '고성'
        else:
            dispname = row['ID']

        # 서대문구, 서귀포시 같이 이름이 3자 이상인 경우에 작은 글자로 표시합니다.
        if len(dispname.splitlines()[-1]) >= 3:
            fontsize, linespacing = 10.0, 1.1
        else:
            fontsize, linespacing = 11, 1.

        annocolor = 'white' if np.abs(row[targetData]) > whitelabelmin else 'black'
        plt.annotate(dispname, (row['x']+0.5, row['y']+0.5), weight='bold', fontsize=fontsize,
                             ha='center', va='center', color=annocolor, linespacing=linespacing)

    # 시도 경계를 그립니다.
    for path in BORDER_LINES:
        ys, xs = zip(*path)
        plt.plot(xs, ys, c='black', lw=2)

    plt.gca().invert_yaxis()
    plt.axis('off')

    cb = plt.colorbar(shrink=.1, aspect=10)
    cb.set_label(datalabel)

    plt.tight_layout()
    plt.show()
```

In [41]:
```
pop.head()
```

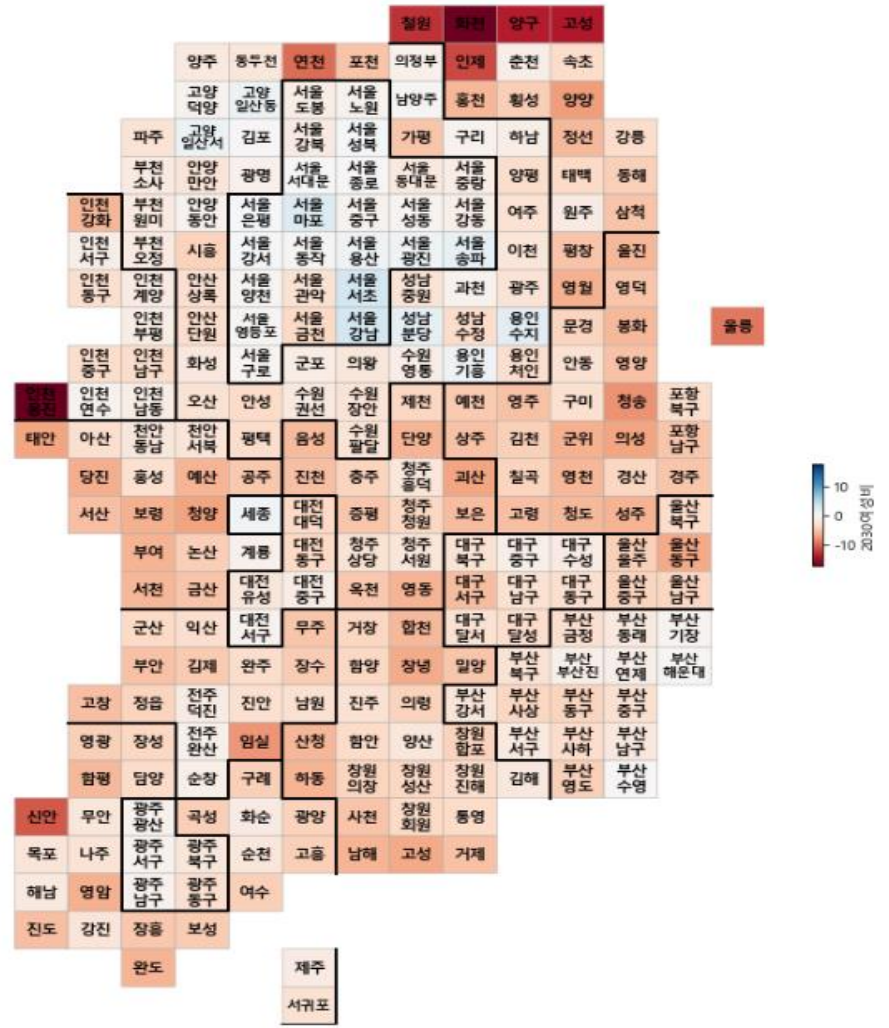
Out[41]:

	광역시도	시도	20-39세여자	20-39세합계	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역	ID	y	x
0	강원도	강릉시	23098.0	49384.0	37679.0	106231.0	107615.0	213846.0	1.226041	0	강릉	3	11
1	강원도	고성군	2529.0	7023.0	7151.0	15899.0	14215.0	30114.0	0.707314	1	고성(강원)	0	10
2	강원도	동해시	9753.0	21264.0	15124.0	47166.0	46131.0	93297.0	1.289738	0	동해	4	11
3	강원도	삼척시	7115.0	15823.0	14610.0	35253.0	34346.0	69599.0	0.973990	1	삼척	5	11
4	강원도	속초시	8752.0	18708.0	12752.0	40288.0	41505.0	81793.0	1.372647	0	속초	1	10

In [42]:
```
# 여성인구수와 합계를 알고 있기 때문에
# 전체 여성인구수와 합계를 나눈 것에 0.5를 뺐습니다.
# 0이면 여성 인구수가 50%입니다.
# RdBu 옵션을 사용해서 0을 기준으로 좌우가 다른 색상을 갖도록 할 수 있다.
# 파란색으로 갈수록 여성비가 높은 것이고
# 빨간색으로 갈수록 여성비가 낮은 것이다.
pop['여성비'] = (pop['인구수여자']/pop['인구수합계'] - 0.5)*100
drawKorea('여성비', pop, 'RdBu')
```



In [43]:
```
# 20-30대 전체 인구에 대한 20-30대 여성의 비율
pop['2030여성비'] = (pop['20-39세여자']/pop['20-39세합계'] - 0.5)*100
drawKorea('2030여성비', pop, 'RdBu')
```



9. Folium에서 인구 소멸 위기 지역 표현하기

In [44]:
```
# 5-9 Folium에서 인구 소멸 위기 지역 표현하기
```

In [45]:
```
# Folium에서 쉽게 인식하도록 하기 위해 pop 데이터에서 ID컬럼을 index로 설정했습니다.
pop_folium = pop.set_index('ID')
pop_folium.head()
```

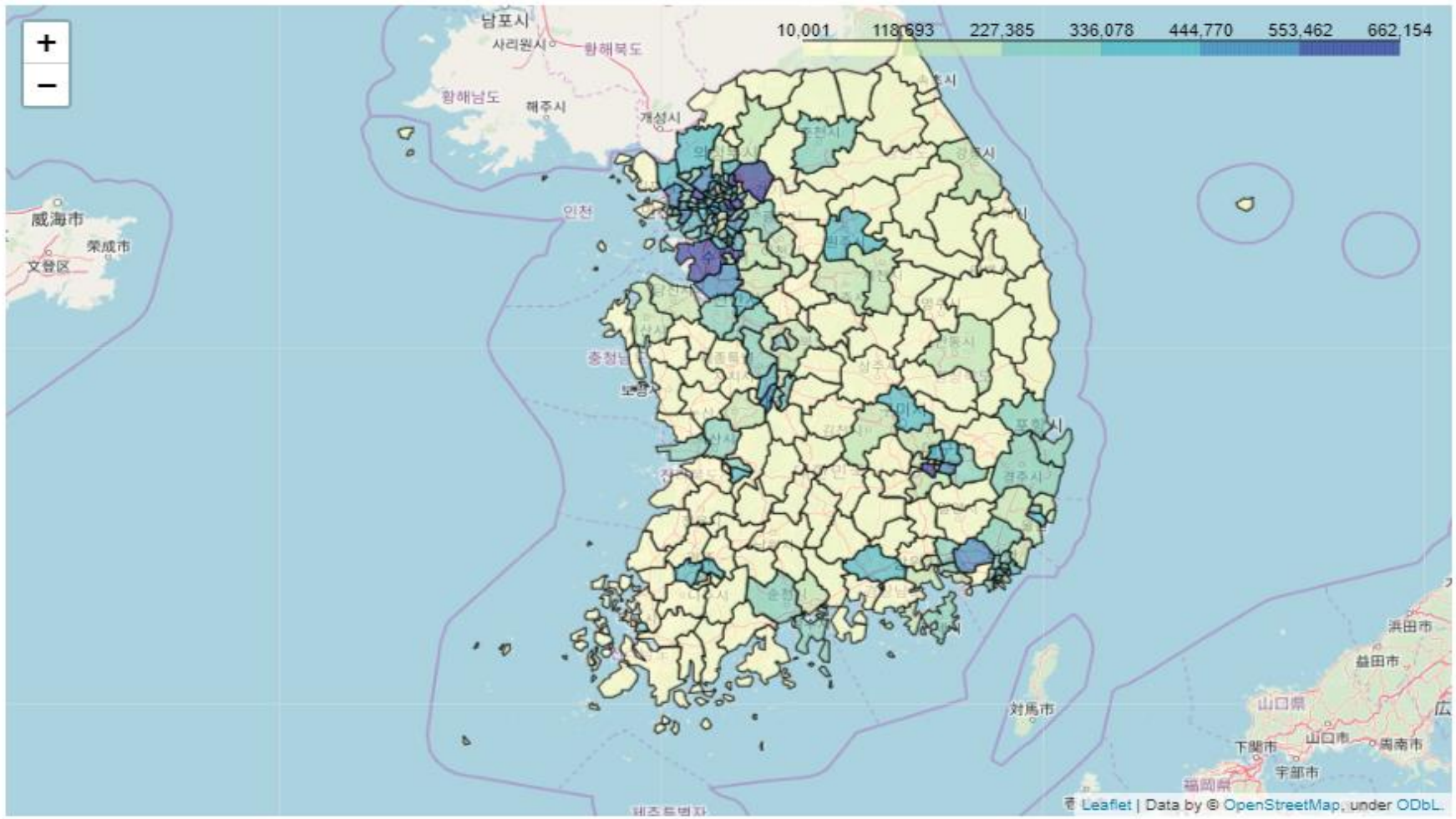
Out[45]:

ID	광역시도	시도	20-39세여자	20-39세합계	65세이상합계	인구수남자	인구수여자	인구수합계	소멸비율	소멸위기지역	y	x	여성비	2030여성비
강릉	강원도	강릉시	23098.0	49384.0	37679.0	106231.0	107615.0	213846.0	1.226041	0	3	11	0.323597	-3.227766
고성(강원)	강원도	고성군	2529.0	7023.0	7151.0	15899.0	14215.0	30114.0	0.707314	1	0	10	-2.796042	-13.989748
동해	강원도	동해시	9753.0	21264.0	15124.0	47166.0	46131.0	93297.0	1.289738	0	4	11	-0.554680	-4.133747
삼척	강원도	삼척시	7115.0	15823.0	14610.0	35253.0	34346.0	69599.0	0.973990	1	5	11	-0.651590	-5.033812
속초	강원도	속초시	8752.0	18708.0	12752.0	40288.0	41505.0	81793.0	1.372647	0	1	10	0.743951	-3.217875

In [46]:
```
# Folium 모듈과 json모듈을 임포트합니다.
import folium
import json
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

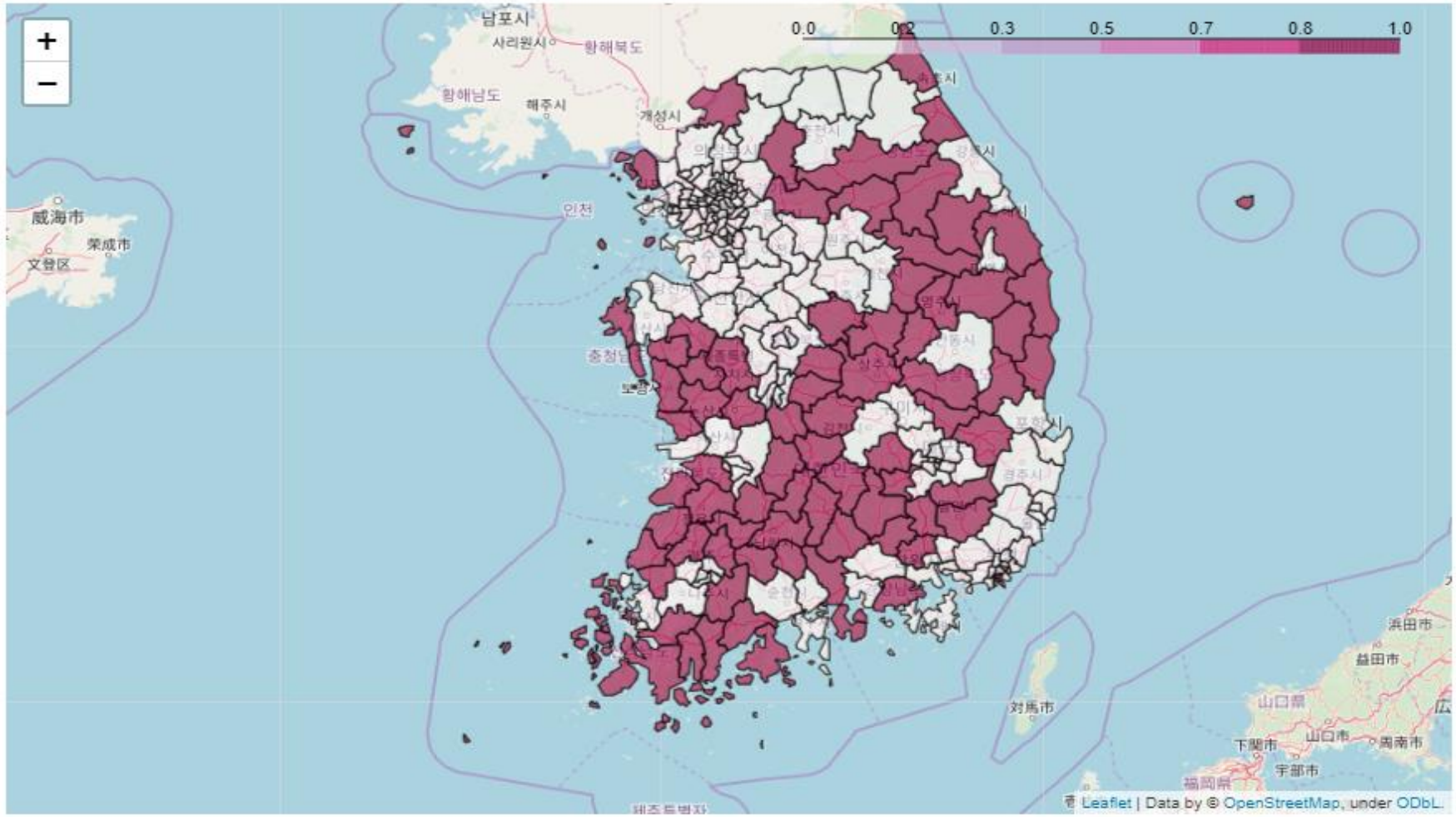
In [47]:
```
# json 파일을 읽은것을 folium을 사용해서
# 지도에 인구수 합계를 표현했습니다.
geo_path = '../data/05. skorea_municipalities_geo_simple.json'
geo_str = json.load(open(geo_path, encoding='utf-8'))

map = folium.Map(location=[36.2002, 127.054], zoom_start=7)
map.choropleth(geo_data = geo_str,
               data = pop_folium['인구수합계'],
               columns = [pop_folium.index, pop_folium['인구수합계']],
               fill_color = 'YlGnBu', #PuRd, YlGnBu
               key_on = 'feature.id')

map
```

Out[47]:



In [48]:
```
# json 파일을 읽은것을 Folium을 사용해서
# 지도에 인구 소멸 위기 지역을 표현했습니다.
map = folium.Map(location=[36.2002, 127.054], zoom_start=7)
map.choropleth(geo_data = geo_str,
               data = pop_folium['소멸위기지역'],
               columns = [pop_folium.index, pop_folium['소멸위기지역']],
               fill_color = 'PuRd', #PuRd, YlGnBu
               key_on = 'feature.id')

map
```

Out[48]:



In [49]:
```
# 지금까지 작업한 데이터를 저장합니다.
draw_korea.to_csv("../data/05. draw_korea.csv", encoding='utf-8', sep=',')
```

10. 소감

앞장에서 실습한 내용의 복습같은 느낌이었습니다.
인구 소멸 위기 지역을 분석한 것을 지도에 시각화해서 보여줌으로써
인구가 감소하고 있다는 수치를 표로 보여주는것보다 훨씬 효과적으로 알아볼 수 있었습니다.

물론 우리가 인구 감소 문제를 어떻게 해결할 순 없겠지만 우리나라의 인구 감소 문제가
점점 심해지고 있다는 것을 알고, 이 문제에 대해 위기감을 가지는 것이 좋겠습니다.